RI5: Delphi Code for Indices Calculator

Šizling, A.L., P. Keil, E. Tjørve, K.M.C. Tjørve, J.D. Žárský, and D. Storch. 202X. Mathematically and biologically consistent framework for presence-absence indices of diversity.

```
unit Indices_Converssion;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes,
Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Math, DN_Sort, DN_Stat,
  Vcl.ExtCtrls, VCLTee.Series, VCLTee.TeEngine, VCLTee.TeeProcs, VCLTee.Chart;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    edK11a: TEdit;
    edK21a: TEdit;
    edK31a: TEdit;
    edL11a: TEdit;
    edL21a: TEdit;
    edL31a: TEdit;
    edI1: TEdit;
    edInterFace: TEdit;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    edI2: TEdit;
    edK11b: TEdit;
    edK21b: TEdit;
    edK31b: TEdit;
    edL11b: TEdit;
    edK22: TEdit;
    edL21b: TEdit;
    edL32: TEdit;
    edL31b: TEdit;
    edK12: TEdit;
    edK32: TEdit;
    edL12: TEdit;
    edL22: TEdit;
    Label9: TLabel;
    Label10: TLabel;
```

```
  Label11: TLabel;
  Label12: TLabel;
  Label13: TLabel;
  Label14: TLabel;
  Label15: TLabel;
  Label16: TLabel;
  Label17: TLabel;
  Label18: TLabel;
  Label19: TLabel;
  Label20: TLabel;
  Label21: TLabel;
  Label22: TLabel;
  Label23: TLabel;
  Label24: TLabel;
  Label25: TLabel;
  Label26: TLabel;
  Label27: TLabel;
  Label28: TLabel;
  Label29: TLabel;
  Label30: TLabel;
  Label31: TLabel;
  butRun: TButton;
  edJsimilarity1: TEdit;
  edSimpson1: TEdit;
  edR1: TEdit;
  edSx1: TEdit;
  edSy1: TEdit;
  edSxy1: TEdit;
  SaveDialog: TSaveDialog;
  OpenDialog: TOpenDialog;
  Label32: TLabel;
  Label33: TLabel;
  Label34: TLabel;
  Label35: TLabel;
  edMeanS: TEdit;
  Label36: TLabel;
  Label37: TLabel;
  Label38: TLabel;
  Label39: TLabel;
  Label40: TLabel;
  edBottomI1: TEdit;
  edUpperI1: TEdit;
  edBottomI2: TEdit;
  edUpperI2: TEdit;
  Label41: TLabel;
  Label42: TLabel;
  edBottomI1a: TEdit;
  edUpperI1a: TEdit;
  edBottomI1b: TEdit;
  edUpperI1b: TEdit;
```

```
    Label43: TLabel;
    Label44: TLabel;
    edErrorI2: TEdit;
    edErrorI1: TEdit;
    Label45: TLabel;
    Label46: TLabel;
    edErrorS: TEdit;
    Label47: TLabel;
    Panel1: TPanel;
    Chart1: TChart;
    Series1: TPointSeries;
    Series2: TLineSeries;
    Chart2: TChart;
    Series3: TPointSeries;
    edText1: TEdit;
    edText2: TEdit;
    edText3: TEdit;
    edJsimilarity2: TEdit;
    edSimpson2: TEdit;
    edR2: TEdit;
    edSx2: TEdit;
    edSy2: TEdit;
    edSxy2: TEdit;
    Series4: TPointSeries;
    Series5: TPointSeries;
    Series6: TPointSeries;
    Series7: TPointSeries;
    Series8: TPointSeries;
    Series9: TPointSeries;
    edInfo: TButton;
    edText4: TEdit;
    Series10: TPointSeries;
    Series11: TPointSeries;
    chckSetAxis: TCheckBox;
    procedure butRunClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure edInfoClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

  rIab,rIa,rIb,rIc: extended;
  rMeanS,rInterface: extended;

  rK1a,rK2a,rK3a,rK1b,rK2b,rK3b,rK1c,rK2c,rK3c: extended;
```

```
  rL1a,rL2a,rL3a,rL1b,rL2b,rL3b,rL1c,rL2c,rL3c: extended;

  rK1a_in,rK2a_in,rK3a_in,rK1b_in,rK2b_in,rK3b_in,rK1c_in,rK2c_in,rK3c_in: extended;
  rL1a_in,rL2a_in,rL3a_in,rL1b_in,rL2b_in,rL3b_in,rL1c_in,rL2c_in,rL3c_in: extended;
  rIab_in,rIc_in,rInterface_in: extended;

  rBottomIab,rBottomIa,rBottomIb,rBottomIc: extended;
  rUpperIab,rUpperIa,rUpperIb,rUpperIc: extended;
  rErrorIab,rErrorIc,rErrorS: extended;

  sX1,sX2,sX3,sX4: string;

  rNula: extended;

  bUninformative_A,bUninformative_B,bUninformative_AB,bUninformative_C: boolean;
  bLze,bSetAxis: boolean;

  sWorkDir: string;
implementation

{$R *.dfm}

uses Unit2;

procedure TForm1.edInfoClick(Sender: TObject);
begin
 Form2.Show();
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
 sWorkDir:=ExtractFilePath(ExpandFileName('pracSoubor'));
//// OpenDialog.InitialDir:=sWorkDir;
 SaveDialog.InitialDir:=sWorkDir;
end;

Procedure Setting();
Var
 rX,rRatio_A,rRatio_B,rRatio_C: extended;
 bIllDefined_A,bIllDefined_B,bIllDefined_C: boolean;
 bIllDefined: boolean;
 sIllDefined: string;
Begin
//rNula:=1E-9;
rNula:=0.001;
bLze:=TRUE;

bSetAxis:=Form1.chckSetAxis.Checked;

rK1a_in:=StrToFloat(Form1.edK11a.Text);
```

```
rK2a_in:=StrToFloat(Form1.edK21a.Text);
rK3a_in:=StrToFloat(Form1.edK31a.Text);

rK1b_in:=StrToFloat(Form1.edK11b.Text);
rK2b_in:=StrToFloat(Form1.edK21b.Text);
rK3b_in:=StrToFloat(Form1.edK31b.Text);

rK1c_in:=StrToFloat(Form1.edK12.Text);
rK2c_in:=StrToFloat(Form1.edK22.Text);
rK3c_in:=StrToFloat(Form1.edK32.Text);

rL1a_in:=StrToFloat(Form1.edL11a.Text);
rL2a_in:=StrToFloat(Form1.edL21a.Text);
rL3a_in:=StrToFloat(Form1.edL31a.Text);

rL1b_in:=StrToFloat(Form1.edL11b.Text);
rL2b_in:=StrToFloat(Form1.edL21b.Text);
rL3b_in:=StrToFloat(Form1.edL31b.Text);

rL1c_in:=StrToFloat(Form1.edL12.Text);
rL2c_in:=StrToFloat(Form1.edL22.Text);
rL3c_in:=StrToFloat(Form1.edL32.Text);

rIab_in:=StrToFloat(Form1.edI1.Text);
rIc_in:=StrToFloat(Form1.edI2.Text);

rInterface_in:=StrToFloat(Form1.edInterface.Text);

rK1a:=rK1a_in; rK2a:=rK2a_in; rK3a:=rK3a_in;
rK1b:=rK1b_in; rK2b:=rK2b_in; rK3b:=rK3b_in;
rK1c:=rK1c_in; rK2c:=rK2c_in; rK3c:=rK3c_in;

rL1a:=rL1a_in; rL2a:=rL2a_in; rL3a:=rL3a_in;
rL1b:=rL1b_in; rL2b:=rL2b_in; rL3b:=rL3b_in;
rL1c:=rL1c_in; rL2c:=rL2c_in; rL3c:=rL3c_in;

rIab:=rIab_in;
rIc:=rIc_in;
rInterface:=rInterface_in;

if Form1.edMeanS.Text='?' then rMeanS:=1 else
rMeanS:=StrToFloat(Form1.edMeanS.Text);

rBottomIab:=StrToFloat(Form1.edBottomI1.Text);
rBottomIa:=StrToFloat(Form1.edBottomI1a.Text);
rBottomIb:=StrToFloat(Form1.edBottomI1b.Text);
rBottomIc:=StrToFloat(Form1.edBottomI2.Text);

rUpperIab:=StrToFloat(Form1.edUpperI1.Text);
rUpperIa:=StrToFloat(Form1.edUpperI1a.Text);
```

```
rUpperIb:=StrToFloat(Form1.edUpperI1b.Text);
rUpperIc:=StrToFloat(Form1.edUpperI2.Text);

rErrorIab:=StrToFloat(Form1.edErrorI1.Text);
rErrorIc:=StrToFloat(Form1.edErrorI2.Text);
rErrorS:=StrToFloat(Form1.edErrorS.Text);

rK1b:=rInterFace*rK1b;
rK2b:=rInterFace*rK2b;
rK3b:=rInterFace*rK3b;

rInterFace:=1;

Form1.edJsimilarity1.Text:='?';
Form1.edJsimilarity2.Text:='?';
Form1.edSimpson1.Text:='?';
Form1.edSimpson2.Text:='?';
Form1.edR1.Text:='?';
Form1.edR2.Text:='?';
Form1.edSx1.Text:='?';
Form1.edSx2.Text:='?';
Form1.edSy1.Text:='?';
Form1.edSy2.Text:='?';
Form1.edSxy1.Text:='?';
Form1.edSxy2.Text:='?';
sX1:=''; sX2:=''; sX3:=''; sX4:='';


bIllDefined_A:=FALSE;
bIllDefined_B:=FALSE;
bIllDefined_C:=FALSE;
if (abs(rL1a)+abs(rL2a)+abs(rL3a)<=rNula) and (abs(rK1a)+abs(rK2a)+abs(rK3a)>rNula)
then bIllDefined_A:=TRUE;
if (abs(rL1b)+abs(rL2b)+abs(rL3b)<=rNula) and (abs(rK1b)+abs(rK2b)+abs(rK3b)>rNula)
then bIllDefined_B:=TRUE;
if (abs(rL1c)+abs(rL2c)+abs(rL3c)<=rNula) and (abs(rK1c)+abs(rK2c)+abs(rK3c)>rNula)
then bIllDefined_C:=TRUE;

if bIllDefined_A or bIllDefined_B or bIllDefined_C then
begin
 sIllDefined:='I cannot solve, for ';
 if bIllDefined_A then sIllDefined:=sIllDefined+'I1a';
 if bIllDefined_B then sIllDefined:=sIllDefined+', I1b';
 if bIllDefined_C then sIllDefined:=sIllDefined+', I2';

 sIllDefined:=sIllDefined+'is/are Ill Defined';

 Form1.edText1.Text:=sIllDefined;
 Form1.Repaint;
```

```
 bLze:=FALSE;
end; //of if bIllDefined_B then

if bLze then
begin

 bUninformative_A:=FALSE;
 bUninformative_B:=FALSE;
 bUninformative_C:=FALSE;
 if (abs(rK1a)+abs(rK2a)+abs(rK3a)<=rNula) then bUninformative_A:=TRUE;
 if (abs(rK1b)+abs(rK2b)+abs(rK3b)<=rNula) then bUninformative_B:=TRUE;
 if (abs(rK1c)+abs(rK2c)+abs(rK3c)<=rNula) then bUninformative_C:=TRUE;

rRatio_A:=0; rRatio_B:=0; rRatio_C:=0;

     if abs(rL1a)>rNula then rRatio_A:=rK1a/rL1a
 else if abs(rL2a)>rNula then rRatio_A:=rK2a/rL2a
 else if abs(rL3a)>rNula then rRatio_A:=rK3a/rL3a;

     if abs(rL1b)>rNula then rRatio_B:=rK1b/rL1b
 else if abs(rL2b)>rNula then rRatio_B:=rK2b/rL2b
 else if abs(rL3b)>rNula then rRatio_B:=rK3b/rL3b;

     if abs(rL1c)>rNula then rRatio_C:=rK1c/rL1c
 else if abs(rL2c)>rNula then rRatio_C:=rK2c/rL2c
 else if abs(rL3c)>rNula then rRatio_C:=rK3c/rL3c;

 if (abs(rK1a-rRatio_A*rL1a)<=rNula) and (abs(rK2a-rRatio_A*rL2a)<=rNula) and
(abs(rK3a-rRatio_A*rL3a)<=rNula) then bUninformative_A:=TRUE;
 if (abs(rK1b-rRatio_B*rL1b)<=rNula) and (abs(rK2b-rRatio_B*rL2b)<=rNula) and
(abs(rK3b-rRatio_B*rL3b)<=rNula) then bUninformative_B:=TRUE;
 if (abs(rK1c-rRatio_C*rL1c)<=rNula) and (abs(rK2c-rRatio_C*rL2c)<=rNula) and
(abs(rK3c-rRatio_C*rL3c)<=rNula) then bUninformative_C:=TRUE;

if bUninformative_A and bUninformative_B and bUninformative_C then
begin
 sIllDefined:='I cannot solve, for none of the indices carry information.';

 Form1.edText1.Text:=sIllDefined;
 Form1.Repaint;

 bLze:=FALSE;
end //of if bUninformative_A then
 else
begin //at least one Index is informative
```

```
bUninformative_AB:=FALSE;
if bUninformative_A and bUninformative_B then bUninformative_AB:=TRUE;

if bUninformative_A and not bUninformative_B then //only B is informative
 begin
  rK1a:=rInterface*rK1b+rRatio_A*rL1b;
  rK2a:=rInterface*rK2b+rRatio_A*rL2b;
  rK3a:=rInterface*rK3b+rRatio_A*rL3b;

  rL1a:=rL1b;
  rL2a:=rL2b;
  rL3a:=rL3b;

  rK1b:=0; rK2b:=0; rK3b:=0;
  rL1b:=1; rL2b:=1; rL3b:=1;

  bUninformative_A:=FALSE;
  bUninformative_B:=TRUE;

  Form1.edK11a.Text:=FloatToStr(rK1a); Form1.edK21a.Text:=FloatToStr(rK2a);
Form1.edK31a.Text:=FloatToStr(rK3a);
  Form1.edL11a.Text:=FloatToStr(rL1a); Form1.edL21a.Text:=FloatToStr(rL2a);
Form1.edL31a.Text:=FloatToStr(rL3a);

  Form1.edK11b.Text:=FloatToStr(rK1b); Form1.edK21b.Text:=FloatToStr(rK2b);
Form1.edK31b.Text:=FloatToStr(rK3b);
  Form1.edL11b.Text:=FloatToStr(rL1b); Form1.edL21b.Text:=FloatToStr(rL2b);
Form1.edL31b.Text:=FloatToStr(rL3b);

  Form1.edInterFace.Text:='+1';
  rInterface:=1;

  Form1.Repaint;

  ShowMessage('Ia carried no information, so I have added up Ia and Ib');
 end //of only B is informative
else if not bUninformative_A and bUninformative_B then //only A is informative
 begin
  rK1a:=rK1a+rInterface*rRatio_B*rL1a;
  rK2a:=rK2a+rInterface*rRatio_B*rL2a;
  rK3a:=rK3a+rInterface*rRatio_B*rL3a;

  rK1b:=0; rK2b:=0; rK3b:=0;
  rL1b:=1; rL2b:=1; rL3b:=1;
```

```
  bUninformative_A:=FALSE;
  bUninformative_B:=TRUE;

  Form1.edK11a.Text:=FloatToStr(rK1a); Form1.edK21a.Text:=FloatToStr(rK2a);
Form1.edK31a.Text:=FloatToStr(rK3a);
  Form1.edL11a.Text:=FloatToStr(rL1a); Form1.edL21a.Text:=FloatToStr(rL2a);
Form1.edL31a.Text:=FloatToStr(rL3a);

  Form1.edK11b.Text:=FloatToStr(rK1b); Form1.edK21b.Text:=FloatToStr(rK2b);
Form1.edK31b.Text:=FloatToStr(rK3b);
  Form1.edL11b.Text:=FloatToStr(rL1b); Form1.edL21b.Text:=FloatToStr(rL2b);
Form1.edL31b.Text:=FloatToStr(rL3b);

  Form1.edInterFace.Text:='+1';
  rInterface:=1;

  Form1.Repaint;

  ShowMessage('Ib carried no information, so I have added up Ia and Ib');
 end; //of only A is informative

end; //of at least one Index is Informative

end; //of if bLze then

End;

Function fDeterminant(var a1,a2,a3,b1,b2,b3,c1,c2,c3: extended): extended;
Var
 Fce: extended;
Begin

Fce:=a1*b2*c3+a2*b3*c1+a3*b1*c2-a1*b3*c2-a2*b1*c3-a3*b2*c1;

fDeterminant:=Fce;
End;

Function bI_depndentData(var rK1a,rK2a,rK3a, rL1a,rL2a,rL3a, rK1b,rK2b,rK3b, rL1b,rL2b,
rL3b: extended; var Ia,Ib: extended; var rNula: extended): boolean;
Var
 a1,a2,a3,b1,b2,b3,c1,c2,c3: extended;
 rDeterminant: extended;

 Fce: boolean;
Begin

 a1:=rK1a-Ia*rL1a; a2:=rK2a-Ia*rL2a; a3:=rK3a-Ia*rL3a;
 b1:=rK1b-Ib*rL1b; b2:=rK2b-Ib*rL2b; b3:=rK3b-Ib*rL3b;
 c1:=0; c2:=0.5; c3:=0.5;
```

```
 rDeterminant:=fDeterminant(a1,a2,a3,b1,b2,b3,c1,c2,c3);

 Fce:=FALSE;
 if abs(rDeterminant)<=rNula then Fce:=TRUE;

bI_depndentData:=Fce;
End;

Function bI_dependent(var rK1a,rK2a,rK3a, rL1a,rL2a,rL3a, rK1b,rK2b,rK3b, rL1b,rL2b,
rL3b: extended; var rNula: extended): boolean;
Var
 a1,a2,a3,b1,b2,b3,c1,c2,c3: extended;
 Sxy,Sx,Sy: extended;
 Ia,Ib: extended;
 rDeterminant: extended;
 Fce: boolean;
Begin
 Sxy:=5;
 Sx:=15;
 Sy:=10;

 Ia:=(rK1a*Sxy+rK2a*Sx+rK3a*Sy)/(rL1a*Sxy+rL2a*Sx+rL3a*Sy);
 Ib:=(rK1b*Sxy+rK2b*Sx+rK3b*Sy)/(rL1b*Sxy+rL2b*Sx+rL3b*Sy);

 Fce:=bI_depndentData(rK1a,rK2a,rK3a, rL1a,rL2a,rL3a, rK1b,rK2b,rK3b, rL1b,rL2b, rL3b,
Ia,Ib, rNula);

 bI_dependent:=Fce;
End;

Procedure ComputeIa(var rIa_final1,rIa_final2: extended; var rIab: extended; var
rBottomLimit_Ia,rUpperLimit_Ia: extended; var
rK1a,rK2a,rK3a,rK1b,rK2b,rK3b,rL1a,rL2a,rL3a,rL1b,rL2b,rL3b,c1,c2,c3: extended; var
Nsolutions: integer);
Var
 rDkk,rDkl,rDlk,rDll: extended;
 rM0,rM1,rM2: extended;
 rDiscriminant,rIaprac: extended;
Begin

rDkk:=fDeterminant(rK1a,rK2a,rK3a,rK1b,rK2b,rK3b,c1,c2,c3);
rDlk:=fDeterminant(rL1a,rL2a,rL3a,rK1b,rK2b,rK3b,c1,c2,c3);
rDkl:=fDeterminant(rK1a,rK2a,rK3a,rL1b,rL2b,rL3b,c1,c2,c3);
rDll:=fDeterminant(rL1a,rL2a,rL3a,rL1b,rL2b,rL3b,c1,c2,c3);

rM2:=rDll;
rM1:=rDlk-rDkl-rIab*rDll;
rM0:=rIab*rDkl-rDkk;
```

```
//solve the quadratic equation
if (abs(rM2)<=rNula) and (abs(rM1)<=rNula) and (abs(rM0)>rNula) then //conflict
 begin
  Nsolutions:=0;
 end
else if (abs(rM2)<=rNula) and (abs(rM1)<=rNula) and (abs(rM0)<=rNula) then //inf solutions
 begin
  rIa_final1:=1;
  rIa_final2:=1;
  Nsolutions:=3;
 end
else if (abs(rM2)<=rNula) and (abs(rM1)>rNula) then //unique solution
 begin
  rIa_final1:=-rM0/rM1;
  rIa_final2:=rIa_final1;
  Nsolutions:=1;
 end
else if abs(rM2)>rNula then
 begin
  rDiscriminant:=rM1*rM1-4*rM2*rM0;

  if rDiscriminant<-rNula then //conflict
   begin
     Nsolutions:=0;
    rDiscriminant:=0;
   end;

  if abs(rDiscriminant)<=rNula then
   begin
    rDiscriminant:=0;
    Nsolutions:=1;
   end;

  if abs(rDiscriminant)>rNula then
   begin
    Nsolutions:=2;
   end;

  rIa_final1:=(-1*rM1+Power(rDiscriminant,0.5))/(2*rM2);
  rIa_final2:=(-1*rM1-Power(rDiscriminant,0.5))/(2*rM2);

  rIaprac:=Min2(rIa_final1,rIa_final2);
  rIa_final2:=Max2(rIa_final1,rIa_final2);
  rIa_final1:=rIaprac;

  if rIa_final2<rBottomIa then
   begin
    Nsolutions:=0;
     rIa_final1:=rBottomIa;
     rIa_final2:=rBottomIa;
```

```
    end
   else if rUpperIa<rIa_final1 then
    begin
     Nsolutions:=0;
     rIa_final1:=rUpperIa;
     rIa_final2:=rUpperIa;
    end
   else if (rBottomIa<=rIa_final1) and (rIa_final2<=rUpperIa) then
    begin
     Nsolutions:=2;
    end
   else if (rIa_final1<rBottomIa) and (rUpperIa<rIa_final2) then
    begin
     Nsolutions:=0;
     rIa_final1:=rBottomIa;
     rIa_final2:=rUpperIa;
    end
   else if (rBottomIa<=rIa_final1) and(rUpperIa<rIa_final2) then
    begin
     Nsolutions:=1;
     rIa_final2:=rIa_final1;
    end
   else if (rIa_final1<rBottomIa) and(rIa_final2<=rUpperIa) then
    begin
     Nsolutions:=1;
     rIa_final1:=rIa_final2;
    end;

 end; //of else if abs(rM2)>rNula then


End;

Function fIafromIc(var rK1c,rK2c,rK3c, rL1c,rL2c,rL3c: extended; var rK1a,rK2a,rK3a,
rL1a,rL2a,rL3a: extended; var rIc: extended; var rNula: extended): extended;
Var
 rXY,rX,rY,rMax: extended;
 rXYabs,rXabs,rYabs: extended;
 Sx,Sy,Sxy: extended;
 Fce: extended;
Begin

rXY:=rIc*rL1c-rK1c;
rX:=rIc*rL2c-rK2c;
rY:=rIc*rL3c-rK3c;

rXYabs:=abs(rXY); rXabs:=abs(rX); rYabs:=abs(rY);

rMax:=Max2(Max2(rXYabs,rXabs),rYabs);
```

```
 if abs(rXYabs-rMax)<=rNula then
  begin
   Sx:=10; Sy:=5;
   Sxy:=-1*(Sx*rX+Sy*rY)/rXY;
  end
 else
 if abs(rXabs-rMax)<=rNula then
  begin
   Sxy:=5; Sy:=10;
   Sx:=-1*(Sxy*rXY+Sy*rY)/rX;
  end
 else
  begin
   Sxy:=5; Sx:=10;
   Sy:=-1*(Sxy*rXY+Sx*rX)/rY;
  end;//of else else if abs(rXY-rMax)<=rNula then


Fce:=(rK1a*Sxy+rK2a*Sx+rK3a*Sy)/(rL1a*Sxy+rL2a*Sx+rL3a*Sy);

fIafromIc:=Fce;
End;

Procedure fIaFromI1(var rI1: extended; var rBottomIa,rUpperIa: extended; var
rK1a,rK2a,rK3a,rK1b,rK2b,rK3b,rL1a,rL2a,rL3a,rL1b,rL2b,rL3b: extended; var Nsolutions:
integer; var rIa_final1,rIa_final2: extended; var rNula: extended);
Var
 rX1,rX2,rX3,rX4,rX5: extended;
 rM0,rM1,rM2: extended;
 rIaprac,rI1prac: extended;
 rDiscriminant: extended;

 bDoIt: boolean;
Begin

bDoIt:=TRUE;
if (abs(rK1a)>rNula) and (abs(rK1b)>rNula) then
 begin
  rX1:=rK2b+rK3b-(rL2b+rL3b)*rI1;
  rX2:=rL1b*rI1-rK1b;
  rX3:=rK2a+rK3a;
  rX4:=rL2a+rL3a;
  rX5:=rL2b+rL3b;

  rM2:=rX5*rL1a-rX4*rL1b;
  rM1:=rX1*rL1a-rX5*rK1a+rX3*rL1b+rX4*rX2;
  rM0:=-1*(rX3*rX2+rX1*rK1a);

 end
else if (abs(rK2a)>rNula) and (abs(rK2b)>rNula) then
```

```
 begin
  rX1:=rK1b+rK3b-(rL1b+rL3b)*rI1;
  rX2:=rL2b*rI1-rK2b;
  rX3:=rK1a+rK3a;
  rX4:=rL1a+rL3a;
  rX5:=rL1b+rL3b;

  rM2:=rX5*rL2a-rX4*rL2b;
  rM1:=rX1*rL2a-rX5*rK2a+rX3*rL2b+rX4*rX2;
  rM0:=-1*(rX3*rX2+rX1*rK2a);
 end
else if (abs(rK3a)>rNula) and (abs(rK3b)>rNula) then
 begin
  rX1:=rK2b+rK1b-(rL2b+rL1b)*rI1;
  rX2:=rL3b*rI1-rK3b;
  rX3:=rK2a+rK1a;
  rX4:=rL2a+rL1a;
  rX5:=rL2b+rL1b;

  rM2:=rX5*rL3a-rX4*rL3b;
  rM1:=rX1*rL3a-rX5*rK3a+rX3*rL3b+rX4*rX2;
  rM0:=-1*(rX3*rX2+rX1*rK3a);
 end
else
 begin
   //nelze
   bDoIt:=FALSE;
 end;

//solve the quadratic equation
if bDoIt then
begin

if (abs(rM2)<=rNula) and (abs(rM1)<=rNula) and (abs(rM0)>rNula) then //conflict
 begin
  Nsolutions:=0;
 end
else if (abs(rM2)<=rNula) and (abs(rM1)<=rNula) and (abs(rM0)<=rNula) then //inf solutions
 begin
  rIa_final1:=1;
  rIa_final2:=1;
  Nsolutions:=3;
 end
else if (abs(rM2)<=rNula) and (abs(rM1)>rNula) then //unique solution
 begin
  rIa_final1:=-rM0/rM1;
  rIa_final2:=rIa_final1;
  Nsolutions:=1;
 end
else if abs(rM2)>rNula then
```

```
begin
 rDiscriminant:=rM1*rM1-4*rM2*rM0;

 if rDiscriminant<-rNula then //conflict
  begin
    Nsolutions:=0;
   rDiscriminant:=0;
  end;

 if abs(rDiscriminant)<=rNula then
  begin
   rDiscriminant:=0;
   Nsolutions:=1;
  end;

 if abs(rDiscriminant)>rNula then
  begin
   Nsolutions:=2;
  end;

 rIa_final1:=(-1*rM1+Power(rDiscriminant,0.5))/(2*rM2);
 rIa_final2:=(-1*rM1-Power(rDiscriminant,0.5))/(2*rM2);

 rIaprac:=Min2(rIa_final1,rIa_final2);
 rIa_final2:=Max2(rIa_final1,rIa_final2);
 rIa_final1:=rIaprac;

 if rIa_final2<rBottomIa then
  begin
   Nsolutions:=0;
    rIa_final1:=rBottomIa;
    rIa_final2:=rBottomIa;
   end
  else if rUpperIa<rIa_final1 then
   begin
    Nsolutions:=0;
    rIa_final1:=rUpperIa;
    rIa_final2:=rUpperIa;
   end
  else if (rBottomIa<=rIa_final1) and (rIa_final2<=rUpperIa) then
   begin
    Nsolutions:=2;
   end
  else if (rIa_final1<rBottomIa) and (rUpperIa<rIa_final2) then
   begin
    Nsolutions:=0;
    rIa_final1:=rBottomIa;
    rIa_final2:=rUpperIa;
   end
  else if (rBottomIa<=rIa_final1) and(rUpperIa<rIa_final2) then
```

```
    begin
     Nsolutions:=1;
     rIa_final2:=rIa_final1;
    end
   else if (rIa_final1<rBottomIa) and(rIa_final2<=rUpperIa) then
    begin
     Nsolutions:=1;
     rIa_final1:=rIa_final2;
    end;



 end; //of else if abs(rM2)>rNula then

end; //of if bDoIt then

End;


Procedure TForm1.butRunClick(Sender: TObject);
Var
 i,iScenario,Nsolutions: integer;

 sNofSolutions: string;

 IA,IB: extended;
rIx1,rIx2,rIy: extended;

  rK1prac,rK2prac,rK3prac, rL1prac,rL2prac,rL3prac: extended;
  rJfinal1,rNfinal1,rRfinal1: extended;
  rJfinal2,rNfinal2,rRfinal2: extended;
  rJfinalX,rNfinalX,rRfinalX: extended;
  rJfinalY,rNfinalY,rRfinalY: extended;
  bI_dependentCJ,bI_dependentCN,bI_dependentCR: boolean;

  aSolution: array [0 .. 4] of extended;
  Nvar: integer;
  sExistence: string;
  bNoSolutionExists: boolean;

  rSxy1,rSx1,rSy1: extended;
  rSxy2,rSx2,rSy2: extended;

  Sx_prac,Sy_prac,Sxy_prac: extended;
  Ia_prac,Ib_prac,Ic_prac,I1_prac: extended;

 rX1,rX2,rX3: extended;

 rK1Aprac,rK2Aprac,rK3Aprac,rK1Bprac,rK2Bprac,rK3Bprac: extended;
 rL1Aprac,rL2Aprac,rL3Aprac,rL1Bprac,rL2Bprac,rL3Bprac: extended;
```

```
 rK1J,rK2J,rK3J,rL1J,rL2J,rL3J: extended;
 rK1N,rK2N,rK3N,rL1N,rL2N,rL3N: extended;
 rK1R,rK2R,rK3R,rL1R,rL2R,rL3R: extended;

 rK1x,rK2x,rK3x,rL1x,rL2x,rL3x: extended;
 rK1y,rK2y,rK3y,rL1y,rL2y,rL3y: extended;

 rDeterminant,rDiscriminant: extended;

 c1,c2,c3: extended;
 rDkk,rDlk,rDkl,rDll: extended;
 rM0,rM1,rM2: extended;

 bSolution1,bSolution2: boolean;
////////////////////////////////////////////////
 M1A,M2A,M3A,M1B,M2B,M3B: extended;

 Cell_Sxy_A,Cell_Sx_A,Cell_Sy_A,RightSide_A: extended;
 Cell_Sxy_B,Cell_Sx_B,Cell_Sy_B,RightSide_B: extended;
 Cell_Sxy_C,Cell_Sx_C,Cell_Sy_C,RightSide_C: extended;

 J,SimNest,R: extended;

 bI_dependentAB,bI_dependentAC,bI_dependentBC: boolean;
 bScenarioA,bIdependent,bConflict: boolean;

 fLS,fRS: TextFile;
Begin
AssignFile(fLS,sWorkDir+'ConvertIndices-fLS.$$$');
AssignFile(fRS,sWorkDir+'ConvertIndices-fRS.$$$');


Setting;

if not bLze then
Begin

   sX1:='There is no solution.';
   sX2:='The reason is that neither I1, or I2 carry information.';
   sX3:='Therefore the indices are not defined properly.';
   sX4:='';

 Form1.edText1.Text:=sX1;
 Form1.edText2.Text:=sX2;
 Form1.edText3.Text:=sX3;
 Form1.edText4.Text:=sX4;
 Form1.RePaint;
End
 else
```

```
Begin //bLze:=TRUE

////////////////////////////////////////////////////////////////////////
bI_dependentAB:=bI_dependent(rK1a,rK2a,rK3a, rL1a,rL2a,rL3a, rK1b,rK2b,rK3b,
rL1b,rL2b,rL3b, rNula);
bI_dependentAC:=bI_dependent(rK1a,rK2a,rK3a, rL1a,rL2a,rL3a, rK1c,rK2c,rK3c,
rL1c,rL2c,rL3c, rNula);
bI_dependentBC:=bI_dependent(rK1b,rK2b,rK3b, rL1b,rL2b,rL3b, rK1c,rK2c,rK3c,
rL1c,rL2c,rL3c, rNula);

if bUninformative_A then
 begin
  bI_dependentAB:=FALSE;
  bI_dependentAC:=FALSE;
 end;

if bUninformative_B then
 begin
  bI_dependentAB:=FALSE;
  bI_dependentBC:=FALSE;
 end;

if bUninformative_C then
 begin
  bI_dependentAC:=FALSE;
  bI_dependentBC:=FALSE;
 end;



//switch #3 to #4
if  not bUninformative_B then //Ib is informative (then also Ia is informative)
begin
 if not bI_dependentAB and bI_dependentAC and not bI_dependentBC then //only Ia,Ic are
dependent
  begin

   rX1:=rK1a; rX2:=rK2a; rX3:=rK3a;
   rK1a:=rK1b; rK2a:=rK2b; rK3a:=rK3b;
   rK1b:=rX1; rK2b:=rX2; rK3b:=rX3;

   rX1:=rL1a; rX2:=rL2a; rX3:=rL3a;
   rL1a:=rL1b; rL2a:=rL2b; rL3a:=rL3b;
   rL1b:=rX1; rL2b:=rX2; rL3b:=rX3;

   bI_dependentAC:=FALSE;
   bI_dependentBC:=TRUE;
 end;
end; //of if  not bUninformative_B
```

```
if not bUninformative_A and not bUninformative_B and not bUninformative_C then //Ia,Ib,Ic are informative
begin

 if not bI_dependentAB and not bI_dependentAC and not bI_dependentBC then //none of Ia,Ib,Ic are i-dependent
  begin
   iScenario:=2;

   rK1x:=rK1a; rK2x:=rK2a; rK3x:=rK3a;
   rL1x:=rL1a; rL2x:=rL2a; rL3x:=rL3a;

   rK1y:=rK1c; rK2y:=rK2c; rK3y:=rK3c;
   rL1y:=rL1c; rL2y:=rL2c; rL3y:=rL3c;

   c1:=rIc*rL1c-rK1c; c2:=rIc*rL2c-rK2c; c3:=rIc*rL3c-rK3c;
   ComputeIa(rIx1,rIx2, rIab, rBottomIa,rUpperIa,
rK1a,rK2a,rK3a,rK1b,rK2b,rK3b,rL1a,rL2a,rL3a,rL1b,rL2b,rL3b,c1,c2,c3,Nsolutions);

   rIy:=rIc;
  end
 else
 if bI_dependentAB and not bI_dependentAC and not bI_dependentBC then  //only Ib and Ic are i-dependent (this includes all cases where Ic,Ia or Ic,Ib only are dependnt)
  begin
  //Ia from Ib, then Ia_final,Ic_final
   if bUninformative_C then iScenario:=1 else iScenario:=2;


   rK1x:=rK1a; rK2x:=rK2a; rK3x:=rK3a;
   rL1x:=rL1a; rL2x:=rL2a; rL3x:=rL3a;

   rK1y:=rK1c; rK2y:=rK2c; rK3y:=rK3c;
   rL1y:=rL1c; rL2y:=rL2c; rL3y:=rL3c;

   //rIa_final is a solution of a quadratic equation
   fIaFromI1(rIab,rBottomIa,rUpperIa,
rK1a,rK2a,rK3a,rK1b,rK2b,rK3b,rL1a,rL2a,rL3a,rL1b,rL2b,rL3b, Nsolutions, rIx1,rIx2, rNula);
   rIy:=rIc;
  end
 else
  if not bI_dependentAB and not bI_dependentAC and bI_dependentBC then //only Ib,Ic are dependent
  begin
   iScenario:=2;

   rIb:=fIafromIc(rK1c,rK2c,rK3c, rL1c,rL2c,rL3c, rK1b,rK2b,rK3b, rL1b,rL2b,rL3b, rIc, rNula);
   rIx1:=rIab-rIb;
```

```
   rIx2:=rIx1;

   rK1x:=rK1a; rK2x:=rK2a; rK3x:=rK3a;
   rL1x:=rL1a; rL2x:=rL2a; rL3x:=rL3a;

   rK1y:=rK1c; rK2y:=rK2c; rK3y:=rK3c;
   rL1y:=rL1c; rL2y:=rL2c; rL3y:=rL3c;

   rIy:=rIc;
  end
 else
  if bI_dependentAB and bI_dependentAC and bI_dependentBC then //all indices are
dependent
  begin
   iScenario:=1;

   rIb:=fIafromIc(rK1c,rK2c,rK3c, rL1c,rL2c,rL3c, rK1b,rK2b,rK3b, rL1b,rL2b,rL3b, rIc,
rNula);
//rIb:=-rIb; //nasilna corekce chyby
   rIx1:=rIab-rIb;
   rIx2:=rIx1;

   rK1x:=rK1a; rK2x:=rK2a; rK3x:=rK3a;
   rL1x:=rL1a; rL2x:=rL2a; rL3x:=rL3a;

   if (rIx1<rBottomIa) or (rUpperIa<rIx1) or ((rIb/rInterface_in)<rBottomIb) or
(rUpperIb<(rIb/rInterface_in)) then
    begin
     Nsolutions:=0;
     sX1:='Index definitions and values are in logical conflict.';
    end
   else
    begin
     Nsolutions:=1;
     sX1:='All Indices are mutually i-dependent.';
    end;

   rK1y:=rK1c; rK2y:=rK2c; rK3y:=rK3c;
   rL1y:=rL1c; rL2y:=rL2c; rL3y:=rL3c;

   rIy:=rIc;
//   rIx2:=rIx1;
  end;

end
 else
if not bUninformative_A and bUninformative_B and bUninformative_C then //only Ia is
informative
begin
   iScenario:=1;
```

```
   rK1x:=rK1a; rK2x:=rK2a; rK3x:=rK3a;
   rL1x:=rL1a; rL2x:=rL2a; rL3x:=rL3a;

   rIx1:=rIab;
   rIx2:=rIx1;
end
 else
if  bUninformative_A and bUninformative_B and not bUninformative_C then //only Ic is
informative
begin
   iScenario:=1;

   rK1x:=rK1c; rK2x:=rK2c; rK3x:=rK3c;
   rL1x:=rL1c; rL2x:=rL2c; rL3x:=rL3c;

   rIx1:=rIc;
   rIx2:=rIx1;
end
 else
if  not bUninformative_A and not bUninformative_B and bUninformative_C then //only Ia
and Ib are informative
begin

 if bI_dependentAB then
  begin
   iScenario:=1;

   fIaFromI1(rIab,rBottomIa,rUpperIa,
rK1a,rK2a,rK3a,rK1b,rK2b,rK3b,rL1a,rL2a,rL3a,rL1b,rL2b,rL3b, Nsolutions, rIx1,rIx2,
rNula);

   rK1x:=rK1a; rK2x:=rK2a; rK3x:=rK3a;
   rL1x:=rL1a; rL2x:=rL2a; rL3x:=rL3a;
  end
 else iScenario:=0;

end
 else
if  not bUninformative_A and bUninformative_B and not bUninformative_C then //only Ia
and Ic are informative
begin

   rK1x:=rK1a; rK2x:=rK2a; rK3x:=rK3a;
   rL1x:=rL1a; rL2x:=rL2a; rL3x:=rL3a;
   rIx1:=rIab;
   rIx2:=rIx1;

   rK1y:=rK1c; rK2y:=rK2c; rK3y:=rK3c;
   rL1y:=rL1c; rL2y:=rL2c; rL3y:=rL3c;
```

```
  rIy:=rIc;


 if bI_dependentAC then //only Ic
  begin
   iScenario:=1;
   Nsolutions:=3;
  end
 else //both Ia and Ic
  begin
   iScenario:=2;
   Nsolutions:=2;
  end;


end;


//PLOT
Form1.Series1.Clear;
Form1.Series2.Clear;
Form1.Series3.Clear;
Form1.Series4.Clear;
Form1.Series5.Clear;
Form1.Series6.Clear;
Form1.Series7.Clear;
Form1.Series8.Clear;
Form1.Series9.Clear;

if bSetAxis then
begin
 With Chart1.LeftAxis do
  begin
   Automatic := false;
   SetMinMax(rBottomIa,rUpperIa);
  end;

 With Chart1.BottomAxis do
  begin
   Automatic := false;
   SetMinMax(rBottomIab,rUpperIab);
  end;

 With Chart2.LeftAxis do
  begin
   Automatic := false;
   SetMinMax(rBottomIa,rUpperIa);
  end;

 With Chart2.BottomAxis do
```

```
  begin
   Automatic := false;
   SetMinMax(rBottomIc,rUpperIc);
 end;
end //of if bSetAxis
 else
begin
 With Chart1.LeftAxis do Automatic := true;
 With Chart1.BottomAxis do Automatic := true;
 With Chart2.LeftAxis do Automatic := true;
 With Chart2.BottomAxis do Automatic := true;
end; //of if bSetAxis

for i:=1 to 10000 do
begin
 Sx_prac:=1; Sy_prac:=random(); Sxy_prac:=random()*Sy_prac;


Ia_prac:=(rK1a*Sxy_prac+rK2a*Sx_prac+rK3a*Sy_prac)/(rL1a*Sxy_prac+rL2a*Sx_prac+r
L3a*Sy_prac);

Ib_prac:=(rK1b*Sxy_prac+rK2b*Sx_prac+rK3b*Sy_prac)/(rL1b*Sxy_prac+rL2b*Sx_prac+r
L3b*Sy_prac);

Ic_prac:=(rK1c*Sxy_prac+rK2c*Sx_prac+rK3c*Sy_prac)/(rL1c*Sxy_prac+rL2c*Sx_prac+r
L3c*Sy_prac);

 I1_prac:=Ia_prac+Ib_prac;


if not bUninformative_C then
begin
  if not (abs(Ic_prac-rIc)<rErrorIc) then
   begin
    Form1.Series1.AddXY(I1_prac,Ia_prac,",clTeeColor);
    Form1.Series3.AddXY(Ic_prac,Ia_prac,",clTeeColor);
   end
  else
   begin
    Form1.Series6.AddXY(I1_prac,Ia_prac,",clTeeColor);
    Form1.Series7.AddXY(Ic_prac,Ia_prac,",clTeeColor);
   end; //of else if not (abs(Ic_prac-rIc)<rErrorIc) then
end
else
begin
  if not (abs(Ia_prac+Ib_prac-rIab)<rErrorIab) then
   begin
    Form1.Series1.AddXY(I1_prac,Ia_prac,",clTeeColor);
    Form1.Series3.AddXY(Ic_prac,Ia_prac,",clTeeColor);
   end
```

```
  else
   begin
    Form1.Series6.AddXY(I1_prac,Ia_prac,'',clTeeColor);
    Form1.Series7.AddXY(Ic_prac,Ia_prac,'',clTeeColor);
   end; //of else if not (abs(Ic_prac-rIc)<rErrorIc) then
end;


end;//of for i:=1 to 1000 do
 Form1.Series8.AddXY(rIab,rIx1,'',clTeeColor);
 Form1.Series8.AddXY(rIab,rIx2,'',clTeeColor);

 Form1.Series2.AddXY(rIab,rBottomIa,'',clTeeColor);
 Form1.Series2.AddXY(rIab,rUpperIa,'',clTeeColor);

 Form1.Series9.AddXY(rIc,rIx1,'',clTeeColor);
 Form1.Series9.AddXY(rIc,rIx2,'',clTeeColor);

if iScenario=1 then
begin
 //convert rIc to J,N or R

 rK1prac:=1;  rK2prac:=0; rK3prac:=0;
 rL1prac:=-1; rL2prac:=1; rL3prac:=1;
 bI_dependentCJ:=bI_dependent(rK1x,rK2x,rK3x, rL1x,rL2x,rL3x,
rK1prac,rK2prac,rK3prac, rL1prac,rL2prac,rL3prac, rNula);
 if bI_dependentCJ then
  begin
   if rL1x*rIx1+rL2x*rIx1+rL3x*rIx1<>0 then
    rJfinalX:=fIafromIc(rK1x,rK2x,rK3x, rL1x,rL2x,rL3x, rK1prac,rK2prac,rK3prac,
rL1prac,rL2prac,rL3prac, rIx1, rNula)
   else rJfinalX:=0;

   if rL1y*rIy+rL2y*rIy+rL3y*rIy<>0 then
    rJfinalY:=fIafromIc(rK1y,rK2y,rK3y, rL1y,rL2y,rL3y, rK1prac,rK2prac,rK3prac,
rL1prac,rL2prac,rL3prac, rIy, rNula)
   else rJfinalY:=rJfinalX;

   bConflict:=FALSE;
   if abs(rJfinalX-rJfinalY)>rNula then bConflict:=TRUE;
   if  (rJfinalX<0) or (1<rJfinalX)  then bConflict:=TRUE;

   if bConflict then
    begin
     sX1:='Index definitions and values are in logical conflict.';
     Nsolutions:=0;
    end
   else
    begin
     sX1:='The indices are i-dependent and do not carry complete information.';
```

```
     sX2:='The indices are equivalent to Jaccard index';
     SX3:='Therefore only Jaccard similarity can be computed.';
     Form1.edJsimilarity1.Text:=FloatToStr(rJfinalX);
    end;

   Form1.edText1.Text:=sX1;
   Form1.edText2.Text:=sX2;
   Form1.edText3.Text:=sX3;
   Form1.Repaint;
  end;

 rK1prac:=1;  rK2prac:=0; rK3prac:=0;
 rL1prac:=0; rL2prac:=0; rL3prac:=1;
 bI_dependentCN:=bI_dependent(rK1x,rK2x,rK3x, rL1x,rL2x,rL3x,
rK1prac,rK2prac,rK3prac, rL1prac,rL2prac,rL3prac, rNula);
 if bI_dependentCN then
  begin
   if rL1x*rIx1+rL2x*rIx1+rL3x*rIx1<>0 then
    rNfinalX:=fIafromIc(rK1x,rK2x,rK3x, rL1x,rL2x,rL3x, rK1prac,rK2prac,rK3prac,
rL1prac,rL2prac,rL3prac, rIx1, rNula)
   else rNfinalX:=0;

   if rL1y*rIy+rL2y*rIy+rL3y*rIy<>0 then
    rNfinalY:=fIafromIc(rK1y,rK2y,rK3y, rL1y,rL2y,rL3y, rK1prac,rK2prac,rK3prac,
rL1prac,rL2prac,rL3prac, rIy, rNula)
   else rNfinalY:=rNfinalX;

   bConflict:=FALSE;
   if abs(rNfinalX-rNfinalY)>rNula then bConflict:=TRUE;
   if  (rNfinalX<0) or (1<rNfinalX)  then bConflict:=TRUE;

   if bConflict then
    begin
     sX1:='Index definitions and values are in logical conflict.';
     Nsolutions:=0;
    end
   else
    begin
     sX1:='The indices are i-dependent and do not carry complete information.';
     sX2:='The indices are equivalent to Simpson index';
     SX3:='Therefore only Simpson index can be computed.';
     Form1.edSimpson1.Text:=FloatToStr(rNfinalX);
    end;

   Form1.edText1.Text:=sX1;
   Form1.edText2.Text:=sX2;
   Form1.edText3.Text:=sX3;
   Form1.Repaint;
  end;
```

```
 rK1prac:=0;  rK2prac:=0; rK3prac:=1;
 rL1prac:=0; rL2prac:=1; rL3prac:=0;
 bI_dependentCR:=bI_dependent(rK1x,rK2x,rK3x, rL1x,rL2x,rL3x,
rK1prac,rK2prac,rK3prac, rL1prac,rL2prac,rL3prac, rNula);
 if bI_dependentCR then
  begin
   if rL1x*rIx1+rL2x*rIx1+rL3x*rIx1<>0 then
    rRfinalX:=fIafromIc(rK1x,rK2x,rK3x, rL1x,rL2x,rL3x, rK1prac,rK2prac,rK3prac,
rL1prac,rL2prac,rL3prac, rIx1, rNula)
   else rRfinalX:=0;

   if rL1y*rIy+rL2y*rIy+rL3y*rIy<>0 then
    rRfinalY:=fIafromIc(rK1y,rK2y,rK3y, rL1y,rL2y,rL3y, rK1prac,rK2prac,rK3prac,
rL1prac,rL2prac,rL3prac, rIy, rNula)
   else rRfinalY:=rRfinalX;

   bConflict:=FALSE;
   if abs(rRfinalX-rRfinalY)>rNula then bConflict:=TRUE;
   if  (rRfinalX<0) or (1<rRfinalX)  then bConflict:=TRUE;

  if bConflict then
      begin
     sX1:='Index definitions and values are in logical conflict.';
     Nsolutions:=0;
    end
   else
    begin
     sX1:='The indices are i-dependent and do not carry complete information.';
     sX2:='The indices are equivalent to Species-Richness gradient';
     SX3:='Therefore only S-R gradient can be computed.';
     Form1.edR1.Text:=FloatToStr(rRfinalX);
    end;

   Form1.edText1.Text:=sX1;
   Form1.edText2.Text:=sX2;
   Form1.edText3.Text:=sX3;
   Form1.Repaint;
  end;

end
else if iScenario=2 then
begin

   Cell_Sxy_A :=rIx1*rL1x-rK1x;
   Cell_Sx_A  :=rIx1*rL2x-rK2x;
   Cell_Sy_A  :=rIx1*rL3x-rK3x;
   RightSide_A:=0;

   Cell_Sxy_B :=rIy*rL1y-rK1y;
   Cell_Sx_B  :=rIy*rL2y-rK2y;
```

```
  Cell_Sy_B  :=rIy*rL3y-rK3y;
  RightSide_B:=0;

  Cell_Sxy_C :=0;
  Cell_Sx_C  :=0.5;
  Cell_Sy_C  :=0.5;
  RightSide_C:=rMeanS;

  ReWrite(fLS);

WriteLn(fLS,FloatToStr(Cell_Sxy_A)+#9+FloatToStr(Cell_Sx_A)+#9+FloatToStr(Cell_Sy_
A) );

WriteLn(fLS,FloatToStr(Cell_Sxy_B)+#9+FloatToStr(Cell_Sx_B)+#9+FloatToStr(Cell_Sy_
B) );

WriteLn(fLS,FloatToStr(Cell_Sxy_C)+#9+FloatToStr(Cell_Sx_C)+#9+FloatToStr(Cell_Sy_
C) );
  CloseFile(fLS);

  ReWrite(fRS);
   WriteLn(fRS,'0');
   WriteLn(fRS,'0');
   WriteLn(fRS,FloatToStr(RightSide_C));
  CloseFile(fRS);

  //SOLVE SYSTEM
  Nvar:=3;
  SystemLinearEquations(fLS,fRS, aSolution, Nvar, sExistence, bNoSolutionExists, rNula,
sWorkDir);
  rSxy1:=aSolution[1];
  rSx1:=aSolution[2];
  rSy1:=aSolution[3];

  Cell_Sxy_A :=rIx2*rL1x-rK1x;
  Cell_Sx_A  :=rIx2*rL2x-rK2x;
  Cell_Sy_A  :=rIx2*rL3x-rK3x;
  RightSide_A:=0;

  ReWrite(fLS);

WriteLn(fLS,FloatToStr(Cell_Sxy_A)+#9+FloatToStr(Cell_Sx_A)+#9+FloatToStr(Cell_Sy_
A) );

WriteLn(fLS,FloatToStr(Cell_Sxy_B)+#9+FloatToStr(Cell_Sx_B)+#9+FloatToStr(Cell_Sy_
B) );

WriteLn(fLS,FloatToStr(Cell_Sxy_C)+#9+FloatToStr(Cell_Sx_C)+#9+FloatToStr(Cell_Sy_
C) );
  CloseFile(fLS);
```

```
 //SOLVE SYSTEM
   Nvar:=3;
   SystemLinearEquations(fLS,fRS, aSolution, Nvar, sExistence, bNoSolutionExists, rNula,
sWorkDir);
   rSxy2:=aSolution[1];
   rSx2:=aSolution[2];
   rSy2:=aSolution[3];

   bSolution1:=FALSE; bSolution2:=FALSE;
   if (0<=rSxy1) and (rSxy1<=rSy1) and (rSy1<=rSx1)  then bSolution1:=TRUE;
   if (0<=rSxy2) and (rSxy2<=rSy2) and (rSy2<=rSx2)  then bSolution2:=TRUE;

   if (rSx1<=rNula) or (rSy1<=rNula) then bSolution1:=FALSE;
   if (rSx2<=rNula) or (rSy2<=rNula) then bSolution2:=FALSE;

   if not bSolution1 and not bSolution2 then
    begin
     Nsolutions:=0;
    end
   else
    begin
     if not bSolution1 then
      begin
       Nsolutions:=1;
       rSxy1:=rSxy2; rSx1:=rSx2; rSy1:=rSy2;
      end;
    end;

   if bSolution1 then
    begin
     rJfinal1:=rSxy1/(-1*rSxy1+rSx1+rSy1);
     rNfinal1:=rSxy1/Min2(rSx1,rSy1);
     rRfinal1:=Min2(rSx1,rSy1)/Max2(rSx1,rSy1);
    end;

   if bSolution2 then
    begin
     rJfinal2:=rSxy2/(-1*rSxy2+rSx2+rSy2);
     rNfinal2:=rSxy2/Min2(rSx2,rSy2);
     rRfinal2:=Min2(rSx2,rSy2)/Max2(rSx2,rSy2);
    end;

    if bSolution1 and ((rJfinal1<0) or (1<rJfinal1) or (rNfinal1<0) or (1<rNfinal1) or
(rRfinal1<0) or (1<rRfinal1)) then bSolution1:=FALSE;
    if bSolution2 and ((rJfinal2<0) or (1<rJfinal2) or (rNfinal2<0) or (1<rNfinal2) or
(rRfinal2<0) or (1<rRfinal2)) then bSolution2:=FALSE;

    if not bSolution1 and bSolution2 then
     begin
```

```
      rSxy1:=rSxy2; rSx1:=rSx2; rSy1:=rSy2;
      rJfinal1:=rJfinal2; rNfinal1:=rNfinal2; rRfinal1:=rRfinal2;
      bSolution1:=TRUE; bSolution2:=FALSE;
     end; //of if not bSolution1 and bSolution2 then

    if (abs(rSxy1-rSxy2)<=rNula) and (abs(rSx1-rSx2)<=rNula) and (abs(rSy1-rSy2)<=rNula)
and (abs(rJfinal1-rJfinal2)<=rNula) and (abs(rNfinal1-rNfinal2)<=rNula) and (abs(rRfinal1-
rRfinal2)<=rNula) then
     begin
      Nsolutions:=1;
     end;

    if (Nsolutions=1) and (bSolution1) then
     begin
      sX1:='There is an unique solution.';
      sX2:='';
      SX3:='';
      sX4:='';
     end //of if Nsolutions=1 then
    else if (Nsolutions=1) and not bSolution1 then
     begin
      Nsolutions:=0;

      if (abs(rIab)<=rNula) and (abs(rIc)<=rNula) then
       begin
        sX1:='The singlar point (I1=I2=0).';
        sX2:='Nonzero index of species-richness gradient is needed,';
        SX3:='to gain a unique solution.';
        sX4:='';

        Form1.edJsimilarity1.Text:='0';
        Form1.edSimpson1.Text:='0';
        Form1.edSxy1.Text:='0';
       end
      else
       begin
        sX1:='There is no solution,';
        sX2:='for index definitions and values are in mutuall conflict.';
        SX3:='';
        sX4:='';
       end;

     end; //of if Nsolutions=1 then

    if bSolution1 and not bSolution2 and (Nsolutions=2) then
     begin
        sX1:='There is an unique solution.';
        sX2:='An ambivalent solution would exist for different index values.';
        SX3:='The reason is the partitioning of I1 into Ia and Ib,';
        sX4:='which results in hump-shapped Ia-I1 relationship.';
```

```
      Nsolutions:=1;
   end //of if bSolution1 and not bSolution2 then
  else if bSolution1 and bSolution2 and (Nsolutions=2) then
   begin
      sX1:='There is an ambivalent solution.';
      SX2:='The reason is the partitioning of I1 into Ia and Ib,';
      sX3:='which results in hump-shapped Ia-I1 relationship.';
      sX4:='';
   end; //of if bSolution1 and bSolution2 then

  rSxy1:=trunc(100*rSxy1+0.5)/100;
  rSx1:=trunc(100*rSx1+0.5)/100;
  rSy1:=trunc(100*rSy1+0.5)/100;

  rJfinal1:=trunc(100*rJfinal1+0.5)/100;
  rNfinal1:=trunc(100*rNfinal1+0.5)/100;
  rRfinal1:=trunc(100*rRfinal1+0.5)/100;

  if Nsolutions=2 then
   begin
    rSxy2:=trunc(100*rSxy2+0.5)/100;
    rSx2:=trunc(100*rSx2+0.5)/100;
    rSy2:=trunc(100*rSy2+0.5)/100;

    rJfinal2:=trunc(100*rJfinal2+0.5)/100;
    rNfinal2:=trunc(100*rNfinal2+0.5)/100;
    rRfinal2:=trunc(100*rRfinal2+0.5)/100;
   end;

  if Nsolutions>0 then
   begin
    Form1.edJsimilarity1.Text:=FloatToStr(rJfinal1);
    Form1.edSimpson1.Text:=FloatToStr(rNfinal1);
    Form1.edR1.Text:=FloatToStr(rRfinal1);

    Form1.edSxy1.Text:=FloatToStr(rSxy1);
    Form1.edSx1.Text:=FloatToStr(rSx1);
    Form1.edSy1.Text:=FloatToStr(rSy1);
   end;

  if Nsolutions=2 then
   begin

    Form1.edJsimilarity2.Text:=FloatToStr(rJfinal2);
    Form1.edSimpson2.Text:=FloatToStr(rNfinal2);
    Form1.edR2.Text:=FloatToStr(rRfinal2);

    Form1.edSxy2.Text:=FloatToStr(rSxy2);
    Form1.edSx2.Text:=FloatToStr(rSx2);
    Form1.edSy2.Text:=FloatToStr(rSy2);
```

```
    end;

   Form1.edText1.Text:=sX1;
   Form1.edText2.Text:=sX2;
   Form1.edText3.Text:=sX3;
   Form1.edText4.Text:=sX4;
   Form1.Repaint;

end
else
begin

 if bUninformative_C and not bUninformative_A  then
  begin
   sX1:='There is infinite number of solutions. No inferrence is possible.';
   sX2:='The reason is that I2 carries no information, and ';
   sX3:='Ia,Ic are i-independent, so I1 carries less information than';
   sX4:='Ia or Ib separatelly.';
  end
 else
  begin
   sX1:='There is no solution.';
   sX2:='The reason is that neither I1, or I2 carry information.';
   sX3:='Therefore the indices are not defined properly.';
   sX4:='';
  end;

 Form1.edText1.Text:=sX1;
 Form1.edText2.Text:=sX2;
 Form1.edText3.Text:=sX3;
 Form1.edText4.Text:=sX4;
 Form1.RePaint;
end;
////////////////////////////////////////////////////////////////////////////////END
end; //bLze=TRUE

DeleteFile(sWorkDir+'ConvertIndices-fLS.$$$');
DeleteFile(sWorkDir+'ConvertIndices-fRS.$$$');
End;

end.
```